Searching CODES Table

Improved logic

Sun, Apr 14, 2002

In the PowerPC systems, nonvolatile memory access time is about 1 μs, which is very slow for a 233 MHz cpu. Searching the CODES table entries is one particular case that happens often. Each 15 Hz cycle, when the local applications are run, a search of CODES must be done to find the program execution pointer. This happens for every entry in the local applications table, LATBL, since each entry contains a 4-character program name, which by convention is preceded with '`LOOP`' to form the 8-character file name that refers to a specific CODES table entry.

An idea for improving the search efficiency is to use the name table scheme that is used for looking up a 6-character analog channel name. The name table uses a hashing scheme that depends upon the fact that the name matches one and only one CODES table entry, which is certainly the case. A new type number can be defined to characterize these entries. What is actually placed in the name table is a pointer to the name field in the CODES table entry, which can be used to access any other field in the entry, plus the table entry number and a name type number. A hash of the name in question is made that is used to find the initial name table entry to be checked. If that entry's pointer points to a matching name, the CODES entry is found immediately. But if it does not match, then another entry is checked, whose relative relation to the first one also depends upon the hash code. If that fails, the same offset is used for the next one to checked, etc. The entire name table is defined to hold as many as 8K entries, so 64K bytes are used. The name table is located at addresses `0xD0000—0xDFFFF` in low memory.

To make this new approach work, there must be a scan of all CODES table entries at reset time to populate the name table, which is in volatile memory. In addition, there must be a removal and add operation whenever a new CODES table entry is formed. This should occur when a program file is copied from another node. The code that does this is in the setting support for the listype that is used for this purpose, maybe `SetProg`.

There are listypes that use idents that are formed from an 8-character program file name. The support for these listypes also involves searching the CODES table, so it could benefit from such a faster search method, too.

Another table for which searching efficiency might be improved by using a similar approach is the DSTRM table, which also has 8-character names. But it is not searched often, so it may not be needed. Also, it's entries are usually populated using the Memory Dump page, so it would be necessary to repeat installation of its entries into the name table periodically to discover any changes that may have been made.